2017

Performance of Message Queue Telemetry Transport Protocol and Constrained Application Protocol in Wireless Sensor Networks

Yaguang Chen
University of Mississippi. Sally McDonnell Barksdale Honors College

PERFORMANCE OF MESSAGE QUEUE TELEMETRY TRANSPORT PROTOCOL AND CONSTRAINED APPLICATION PROTOCOL IN WIRELESS SENSOR NETWORKS

by
Yaquang Chen

A thesis submitted to the faculty of The University of Mississippi in partial fulfillment of the requirements of the Sally McDonnell Barksdale Honors College.

Oxford
May 2017

Approved by

_____

Advisor: Professor Matthew Morrison

_____

Reader: Professor Ramanarayanan "Vish" Viswanathan

_____

Reader: Professor Richard K Gordon

ACKNOWLEDGEMENTS

After an intensive period of seven months, today is the day: writing this note of thanks is the finishing touch of my thesis.

I would first like to express my sincere thanks to the Honors College and Department of Electrical Engineering for giving me this thesis and providing me with all the necessary facilities to accomplish the thesis. It has been a period of intense learning for me, not only in the scientific area, but also on a personal level.

I would like to thank my thesis advisor, Dr. Matthew Morrison for his valuable guidance. Dr. Morrison provided me with support that I needed to choose the right direction and successfully complete my thesis. I would like to thank to Dr. Ramanarayanan "Vish" Viswanathan and Dr. Richard Gordon for being my readers and giving all the valuable suggestions.

I would like to thank my Senior-design team members, Maisha Sadia and Xavir Pittman, who have also put countless ideas and effort into this project. This thesis would not reach the prospective without their hard work.

I would also like to thank my parents for their wise counsel and sympathetic ear. Without their love and support, I would not have any chance to get a higher education and finish a thesis.

ABSTRACT

The Wireless Sensor Networks (WSNs) are formed by a broker/server, a few gateways and a large numbers of Sensor/Actuator (SA) devices, which are battery-operated and able to collect information about their environment, storage a small amount of data and transfer those data to the gateways for further operations. In this paper, two potential high performance protocols that could be implemented for WSNs are studied. These two protocols are MQTT-SN and CoAP protocols and their performances of time efficiency and power efficiency are mainly discussed and studied. A set of principles and designed experiments for both protocols are presented. The simulation results are generated and MQTT-SN protocol shows a better performance in time efficiency of transmitting data comparing to CoAP protocol. Additionally, the experiment on Power Consumption of MQTT-SN protocol and CoAP need to be improved and re-preformed before the more power-efficient protocol can be decided.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Wireless Sensor Networks (WSNs) have played an increasingly important role in industrial automation, environment monitoring, and transportation business, which has resulted in increased interest from industrial and research perspectives." The increased proliferation of WSNs may be attributed to their availabilities of inexpensive, low energy consumption and simplicity to deploy [1].

The WSNs are formed by a broker/server, a few gateways and a large numbers of Sensor/Actuator (SA) devices, which are battery-operated and able to collect information about their environment, storage a small amount of data and transfer those data to the gateways for further operations [2]. A WSN is comprised of two subnets: One subnet consists of sensor nodes and one or more gateways, which will receive data from sensor nodes via WSN protocols. The other subnet will connect the gateway to a broker/server and deliver the data that gateway has collected from sensor nodes. After that, any client who needs the sensor data can connect to the broker/server and receive the data [3].

Such as the situation of low-power wearable sensor devices X-patch [4] designed by the X2 Biosystems Inc., Xpatch requires the feature of real time communication and low power consumption [4]. Due to these features of sensor devices, efficient transfer of sensor data to a server using WSNs requires a protocol which is transmission efficient and energy efficient. Additionally, Sensor/Actuator (SA) devices inside of WSNs should have various network addresses at any time and SA nodes are possible to fail at any time. With regards to traditional networks (e.g. Internet, LAN, enterprise network, etc) which use network addresses as

communication, their dynamic and temporal nature would constitute troubles and failures in the WSNs. However, a data-centric communication using Machine-to-Machine (M2M) protocol does not require the address or identity of the devices but only interested in the content of the data. In this approach, the SA maintains and manages communication of data between each other for different purposes [6]. In this case, a data-centric communication approach by using M2M protocol is needed. Publish/Subscribe (pub/sub) messaging systems are a well-known data-centric communication technique, and widely used because of their support to a dynamic network typology.

Message Queue Telemetry Transport Protocol for Sensor Network (MQTT-SN) [5] is the open pub/sub protocol which was studied and applied in this thesis. MQTT-S or MQTT-SN is an extension for sensor networks of Message Queuing Telemetry Transport (MQTT), which does not consider the case of SA devices. Both MQTT and MQTT-SN are client-server protocols, for which a server is needed to distribute message between the client applications. To enable the server to run on machines which do not have capacity for running a JVM, MQTT and MQTT-SN are written in C and sensors and actuators, which are very small and lacking in power, are often the sources and destinations. MQTT and MQTT-SN also use the publish-subscribe paradigm, rather than queueing.

Another considerable protocol would be the Constrained Application Protocol (CoAP). CoAP is a well-developed layer protocol which utilizes the concept of “Internet of Things”(IoT) and is intended to be used in the communication of Machine to Machine (M2M) applications. The idea behind IoT is to connect many devices, or “things”, to many services in the web in an easy and convenient way [6].

The objective of this paper is to decide which protocol, MQTT-SN protocol or CoAP protocol, better satisfied the WSN. The rest of paper will discuss and proceed to experiments separately with respect to transmission efficiency and power consumption efficiency. Descision will be made based on performances of data-transmit efficiency and power consumption of MQTT-SN protocol with CoAP protocol during a client-server data transmission.

CHAPTER 2
LITERATURE REVIEW

MQTT-SN

MQTT-SN very similar to MQTT which connect SA devices to an MQTT broker, enable a smooth integration of the WSNs and enable a simple and accurate operations of the gateways Comparing to the MQTT, MQTT-SN considers the wireless network constraints such as high link failure rates, low bandwidth and short message payload. The property of network independent allows MQTT-SN to run on any network that provides with point-to-point data transfer service and one-hop broadcast data transfer service. More than that, MQTT-SN protocol is designed for implementation on low-cost and low-power SA devices with limited processing and storage. It makes operations on the SA devices side maintained very simple process [2]. MQTT-SN are also adapted for communication over low bandwidth links which are capable of only short messages and where network interruptions are common [5].

The three mainly types of architectures of MQTT(-SN) are Transparent Gateway architecture, Hybrid Gateway architecture and Aggregated Gateway architecture, as shown in Figure 1. In Transparent Gateway architecture, every sensor device will have one gateway which is connected to the broker/server. There will be as many MQTT connections between Gateways and the broker as there are MQTT-SN sensor devices connected to Gateways. In this type of architecture, the infrastructure cost is high and sensor may often lose connections to Gateway because of the fragile connection. As for Aggregated Gateway architecture, there is only one Gateway which will collect all the sensors data and send them to the broker/server. In this case, it Is hard for a single Gateway to maintain all the connections and transfer data. There will a high possibility of losing data and causing failure of connections. Hybrid Gateway architecture is the

architecture where sensor nodes can be connected to multiple Gateways and the number Gateways are usually less than the number of sensor nodes in the network. Compare to two other architectures, Hybrid is more viable and realizable [6].

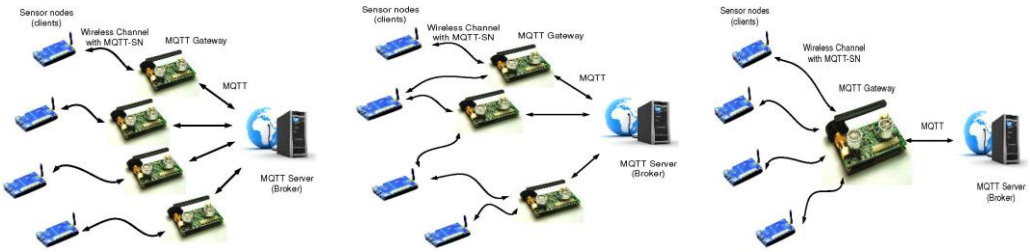

Figure 1. Three mainly types of architectures of MQTT-SN: Transparent Gateway architecture, Hybrid Gateway architecture and Aggregated Gateway architecture

CoAP

CoAP is specifically specialized for use with constrained devices and networks such as low-power networks and lossy networks. CoAP runs on the User Datagram Protocol (UDP) as the transport layer which provide the minimum packet size because UDP only use 8 bytes as packet header. Moreover, CoAP use a similar and a shared communication model with HTTP, which is Representational State Transfer (REST). Since REST has similar operation and the same infrastructure as HTTP, there is no need for a complex translator for CoAP to interoperate with HTTP [7].

CoAP is especially optimized for low-power, low-bandwidth, or “constrained” networks, which makes it a perfect fit for packaging the data collected from a sensor device, such as X-patch. The design features of ease of mapping onto the HTTP protocol make it ideal for constrained networks. Due to having a similar web stack structure, CoAP can exploit the same paradigms that are used in http. This allows it to connect to an http server via a proxy, therefore

allowing the data in the constrained network to be accessed from the Internet, or "Big Data". The similarity in structure are illustrated in the diagram below. Instead of using the more robust web representations such as IPv4, TLS/TCP transport, and XML/JSON, CoAP uses more lightweight substitutes such as 6LoWPAN, DTLS/UDP and Binary Web Objects respectively. This highly increases efficiency in a low power network as well.

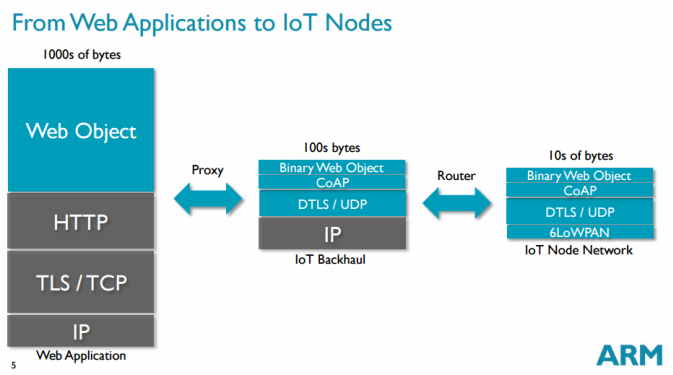

Figure 2: From Web Application to IoT Nodes

CoAP uses a RESTful web architecture just like HTTP. This means that in CoAP, web resources are resolved, identified, and manipulated in the same way as in HTTP. The REST architecture allows this by using the built in features of any given protocol. For example, CoAP uses URI's that can map directly to HTTP and also uses the four request/ response methods: GET, POST, PUT, DELETE in RESTful semantics, which corresponds to HTTP as well.

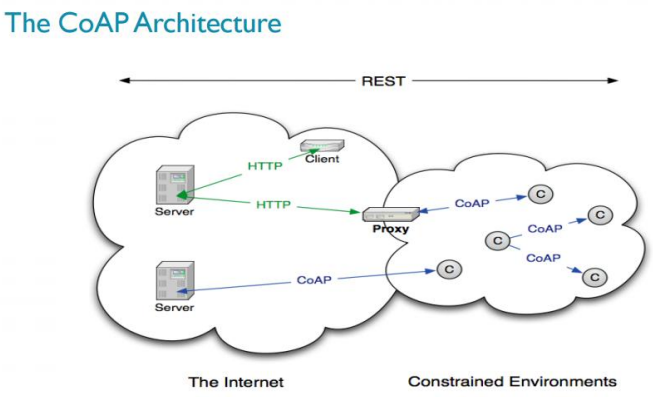

Figure 3: The CoAP Architecture

Asynchronous Network: A Synchronous network such as HTTP uses TCP which is very connection oriented. Multiple requests have to be sent to the server even for simple information, and exchanging information in both directions in discrete times is also difficult due to need of keeping an open connection. This makes HTTP quite an inflexible protocol for IoT applications [11]. CoAP on the other hand uses UDP transfer protocol which is completely asynchronous. This allows features such as subscriptions and multicast group communications. Multicast is particularly useful for low power devices and networks as it allows for a single request to be transmitted to many receiving devices, or nodes (CoAP-CoAP peers) [12]. The diagram below illustrates this by showing how this can be applied to smart lighting.

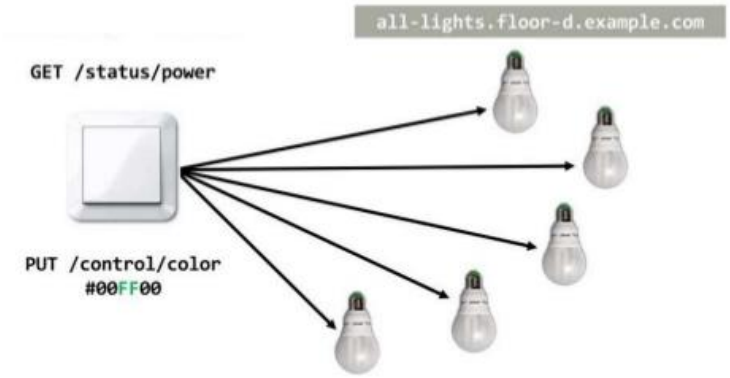

Figure 4 [19]: Example of CoAP Application in Smart Lighting

There currently exist research studies that have already discussed some performance comparisons between MQTT-SN and CoAP. One research [2] discussed the similarity and major difference between them: both protocols are proposed for resource-constrained devices, but they use different transport layer - MQTT(-SN) runs on the Transmission Control Protocol (TCP)

while CoAP runs on the User Datagram Protocol (UDP). TCP is a highly reliable host-to-host protocol which is connection-oriented, end-to-end and designed to fit into a layered hierarchy of protocols and support multi-network applications [8]. Different from TCP, UDP is defined to make a datagram mode available for the packet-switched computer while communicating in an interconnected set of computer networks. UDP provides a simple procedure to transmit data with a minimum of protocol mechanism [9]. Research also compared their performance of the delay due to the loss rate by applying a common middleware. The result of this experiment shows that when the loss rate is lower, MQTT performs better than CoAP with delay of less than one second for every message. But at 25% loss rate or higher, CoAP has lower delays than MQTT(-SN) [10].

CHAPTER 3
EXPERIMENT SETUP

The objective is to compare the client-side performances of data-transmit efficiency and power consumption of MQTT-SN protocol with CoAP protocol during a data transmission between a client and a server. To achieve the goal of comparing the two protocols' two different features, experiment are divided into two subparts: experiment of data-transmit efficiency and experiment of power consumption measurement. MQTT-SN will be performed firstly and CoAP secondly in each subpart. Both experiments of both protocols have to run and test on the same device, which is Raspberry Pi 3 B (Pi3) connected to the network through Local Area Network Ethernet cable. During the Simulation Procedure, the Pi3 has been configured as a client device which will send data to the server with MQTT-SN protocol first and CoAP protocols second. On the other end, a Linux environment computer are serving as the server that will receive sending data from the client (Pi3).

3.1 Simulation Setup

MQTT-SN Setup

As mentioned above, MQTT-SN transmits data through a broker where it can both receive published data and send them to any client who has subscribed to enable a smooth integration of the WSNs. In this case, the test was implemented using Mosquitto broker. Mosquitto is one of the most popular and stable brokers, and provides a lightweight server implementation of the MQTT-SN protocol. Lightweight means that only necessary functions are included and functions are coded as efficiently as possible [13]. Mosquitto can also translate and transfer messages

between MQTT and MQTT-SN acting as a gateway when devices communicate with either protocols. Additionally, Mosquitto can be easily installed on the Raspberry Pi. To install and test Mosquitto on the Raspberry Pi, follow this tutorial [14].

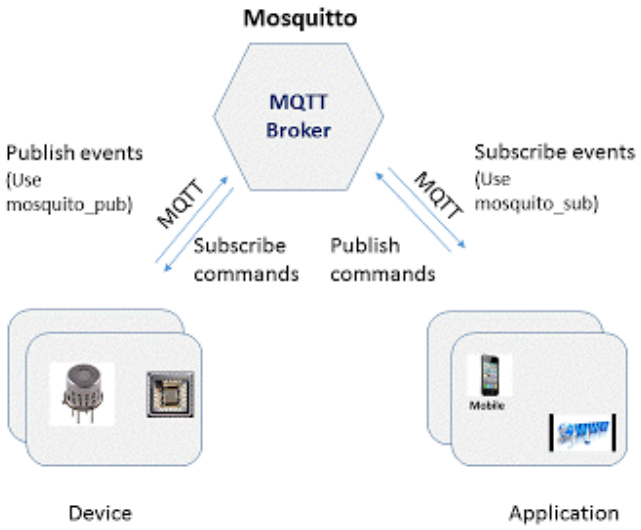

Figure 5: Functionablity of Mosquitto in MQTT(-SN) protocol

CoAP Setup

During the CoAP test, server and clients are implemented using txThings. Txthings is a simple library for CoAP and runs based on Twisted which is an asynchronous I/O framework and networking engine written in Python. To emulate similar environment for CoAP that only involve client and server, the Raspberry Pi was set as the client, while the Linux environment PC was set as the server. To install and run CoAP on Raspberry Pi and the PC, we followed this tutorial [15].

Wireshark setup

In the experiment of comparing Time Efficiency of two protocols, Wireshark software is installed on server side to measure the data transmission time. Wireshark is an open source packet analyzer and is used for network troubleshooting and analysis. WIreshark provides network interface controllers where users can see all traffic in progress respect to a time stamp. Thus, the time of each protocol used to receive different size of data file can be obtained by the reading of time period between starting receiving data and finishing receiving data. Wireshark can be easily installed and tested by following this tutorial [16].

Power Consumption

As for the power consumption of the two protocols, rather than measuring it directly, an indirectly method is applied by using the formula (1):

$$P=UI \quad (1)$$

where P is the power consumption of the Raspberry Pi with unit in Watt, U is the voltage across the Raspberry Pi with unit in Volt and I is the current through the Raspberry unit in Amp. In the designed circuit, 5-Volt Raspberry Pi is series with a 1.5 Ohm resistor connecting to an Oscilloscope. The current of Raspberry Pi could be obtained by applying the Ohm's law (2) on the 1.5 Ohm resistor:

$$I = V/R \quad (2)$$

where I is the current through the resistor in Amp, V is the voltage across the resistor in Volt measured by the Oscilloscope, and R is the resistance of the resistor in Ohm. The circuit is connected as Figure 6:

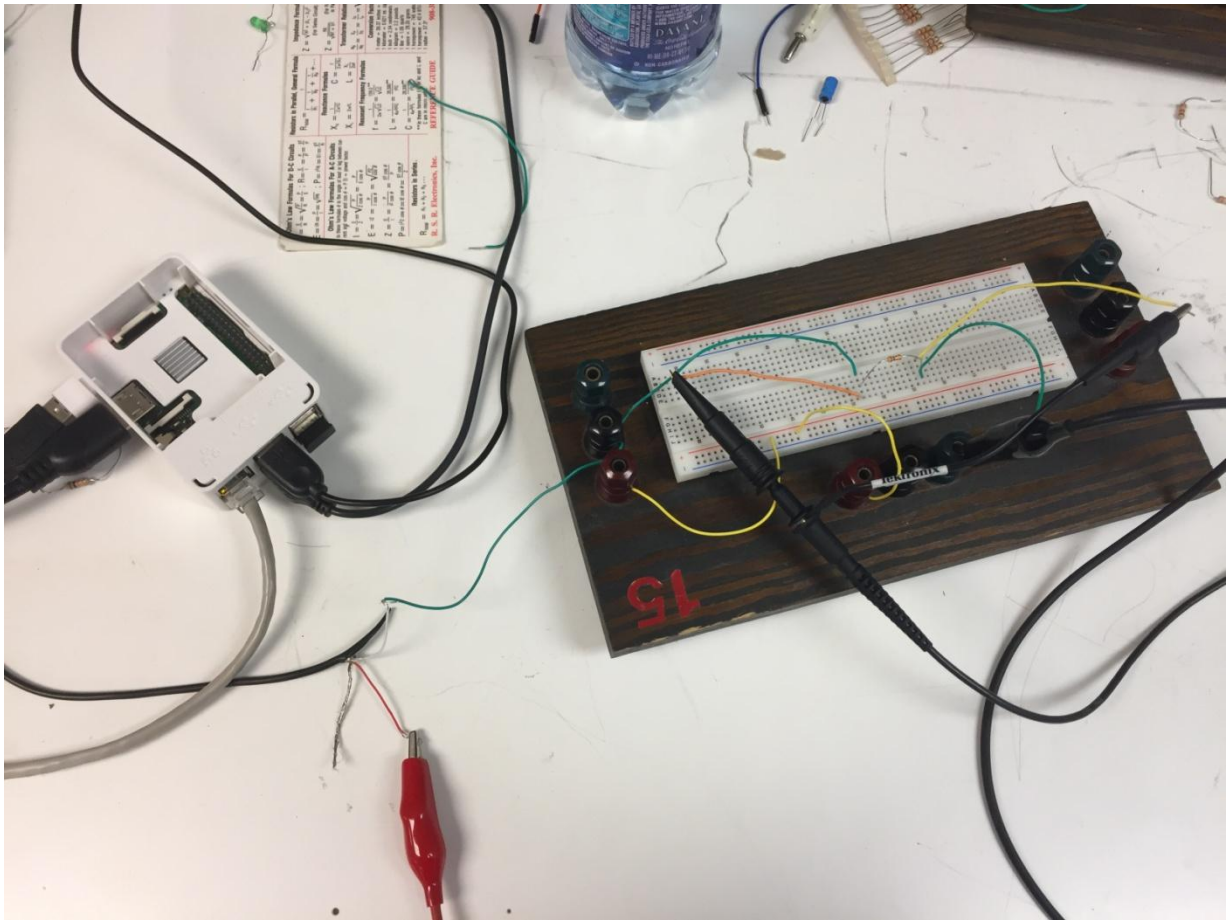

Figure 6: Circuit used to obtain the power consumption of Raspberry Pi.

3.2 Experiment Procedure

Once have Mosquitto of MQTT-SN protocol and txThings of CoAP successfully installed tested on both Raspberry Pi and PC, the experiment can start with running MQTT-SN Mosquitto-subscriber (server) on the Linux computer. The Mosquitto-subscriber will subscribe

the IP address of the client Raspberry Pi and wait for the it publishing data. The Wireshark should be started capturing the traffic in progress on the server as well. Then, Mosquitto-publisher can be running on the Raspberry Pi (client) and publishing data file. As soon as the server subscribes the client and starts receiving the data published by the client, this transmission will appear on the Wireshark interface with information of time, source IP, destination IP, protocol type, length of data and information of this connection. When transmission is over, there will be a notice that the server completes receiving published data and Wireshark can be stopped capturing at this moment.

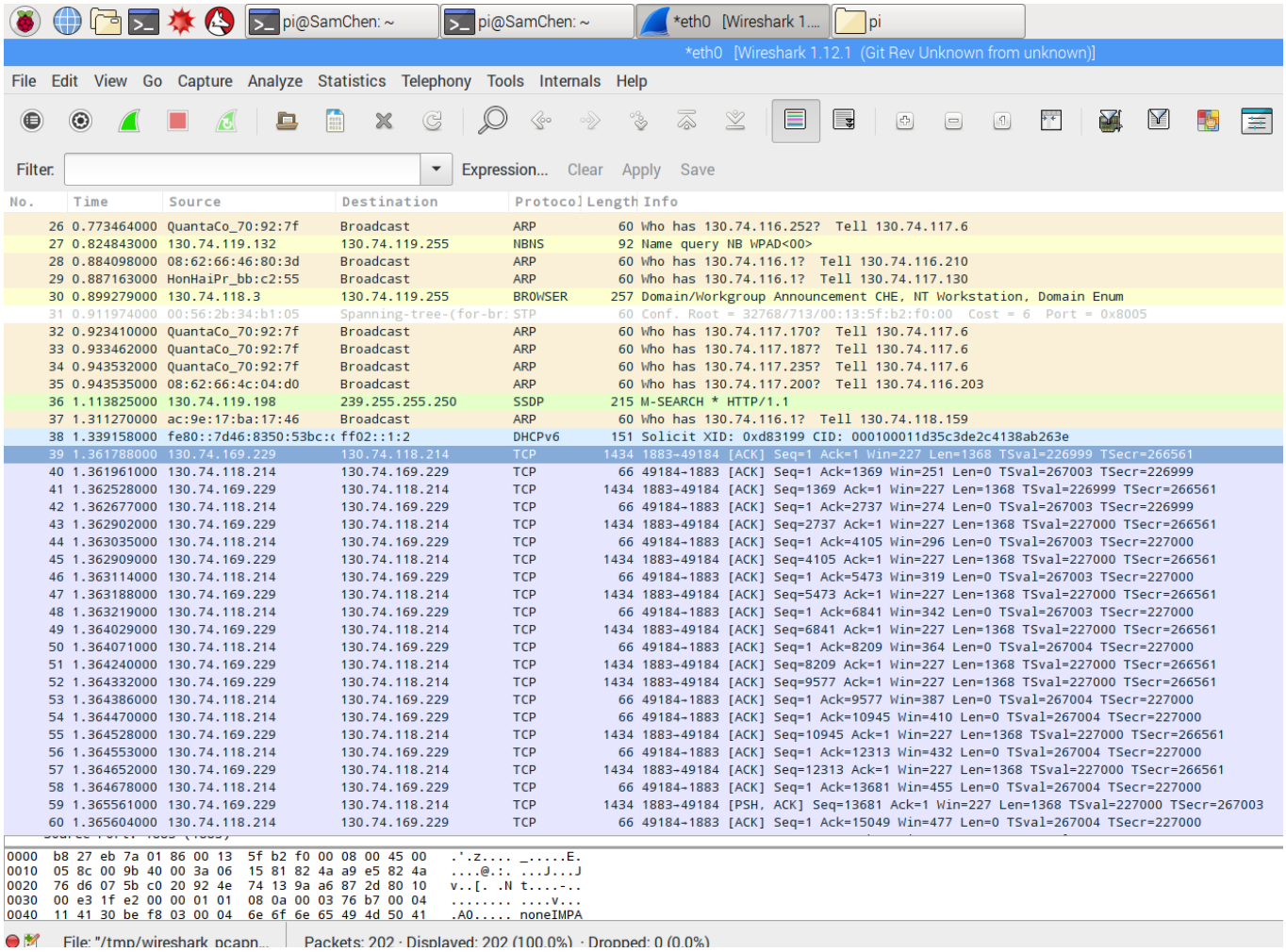

Figure 7: Example of Wireshark's interface when Moquitto server started receiving 63 Kbit text file.

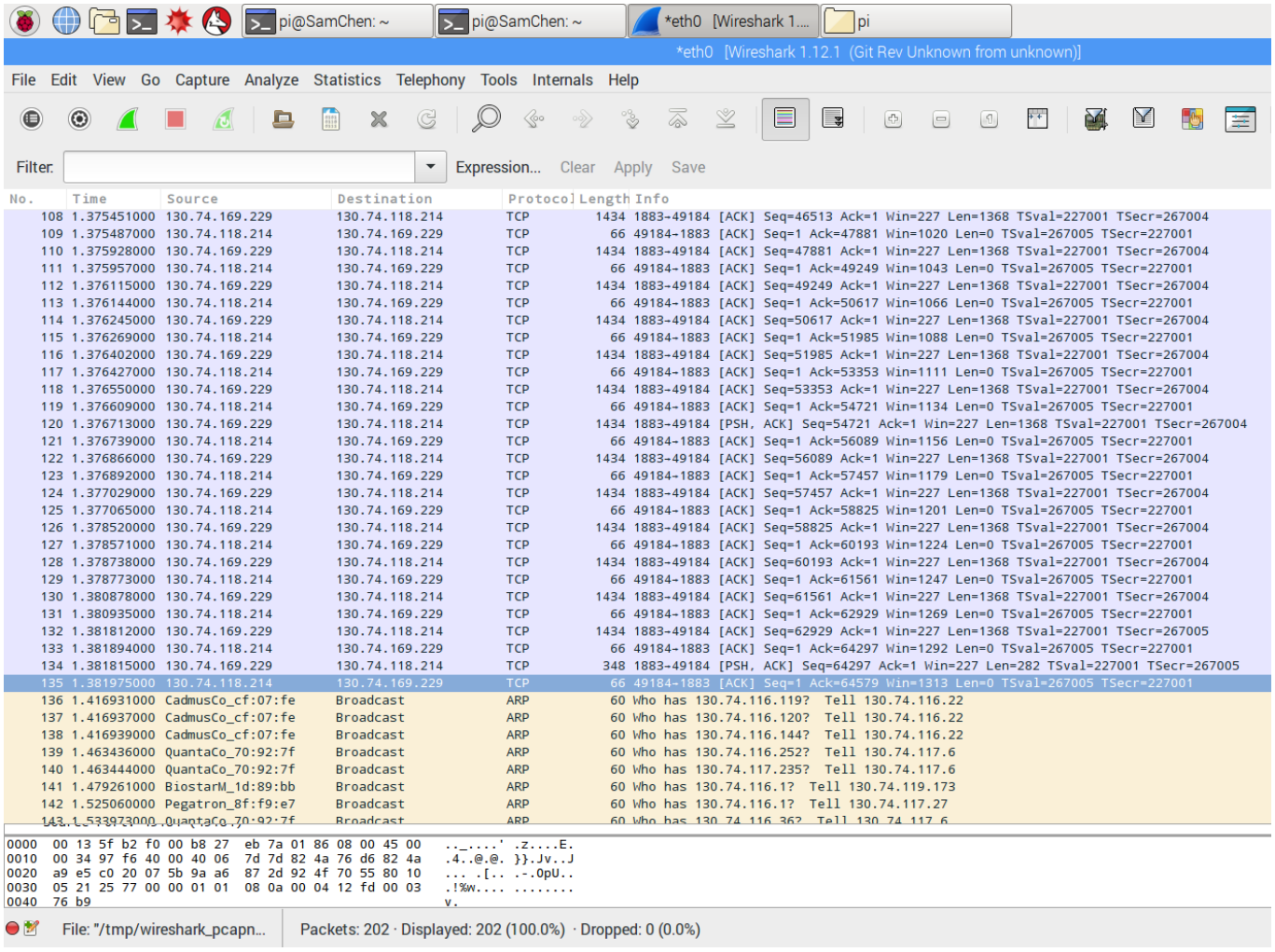

Figure 8: Example of Wireshark's interface when Moquitto server finished receiving 63 Kbit text.

Similarly for CoAP, it first has to run the CoAP server on the PC with calling the ClientGET.py file and start the capture of Wireshark on the PC. Then the CoAP client can be started with calling ClientPUT.py file where it included function to transmit different data file. Once the CoAP server has built connection with the CoAP client and starts receiving data sent by the client, transmission information will be visible on the Wireshark's capture interface as well. The capture of Wireshark can be stopped when data transmission is over.

Power Consumption

To perform this experiment, the circuit is connected as mentioned above where the client Raspberry Pi is in series with a 1.5 Ohm resistor measured by an Oscilloscope. When the circuit is built up, repeat the experiment procedure described in Time Efficiency experiment with MQTT-SN and CoAP protocols except running Wireshark which is not used here. There will be a changing waveform signal on the Oscilloscope screen while transmitting the data file with both protocols. Use the Oscilloscope to save the signal and calculate the mean value in the period of transmission. The mean value will be used to calculate the current through Raspberry Pi with formula (1) and then to obtain the power consumption of Raspberry Pi with formula (2).

CHAPTER 4
SIMULATION RESULTS

4.1 Time Efficiency Experiment

In this experiment, totally 7 different size of text file of data were transmitted, which are 2.25Kbit, 4.5 Kbit, 6.76 Kbit, 9.02 Kbit, 11.2 Kbit, 31.4 Kbit and 63 Kbit. The time consumed in each transmission is different based on the size of the data file. The following table and graph provide the results of time consuming in both MQTT-SN and CoAP protocols versus sizes of data file. The consumed time, which is the take taken in the table, is calculated by subtracting start time from end time.

	data size/ Kb	entries	start time/seconds	end time/seconds	time taken / seconds
	0	0	0	0	0
	2.25	35	1.132725	1.133393	0.000668
MQTT	4.5	70	5.469954	5.470947	0.000993
	6.76	115	2.053459	2.055486	0.002027
	9.02	140	2.680214	2.682535	0.002321
	11.2	175	2.471589	2.4751	0.003511
	31.4	500	2.810536	2.821854	0.011318
	63	1000	1.361788	1.381975	0.020187
	data size/ Kb	entries	start time/seconds	end time/seconds	time taken / seconds
	0	0	0	0	0
	2.25	35	6.194207	7.22632	1.032113
CoAP	4.5	70	5.967338	7.772799	1.805461
	6.76	115	6.063139	8.647908	2.584769
	9.02	140	5.776667	9.746863	3.970196
	11.2	175	5.988222	10.619595	4.631373
	31.4	500	6.438516	19.146653	12.708137
	63	1000	5.646544	30.783964	25.13742

Table 1: Time consuming in MQTT-SN and CoAP protocols with different data file transmitted.

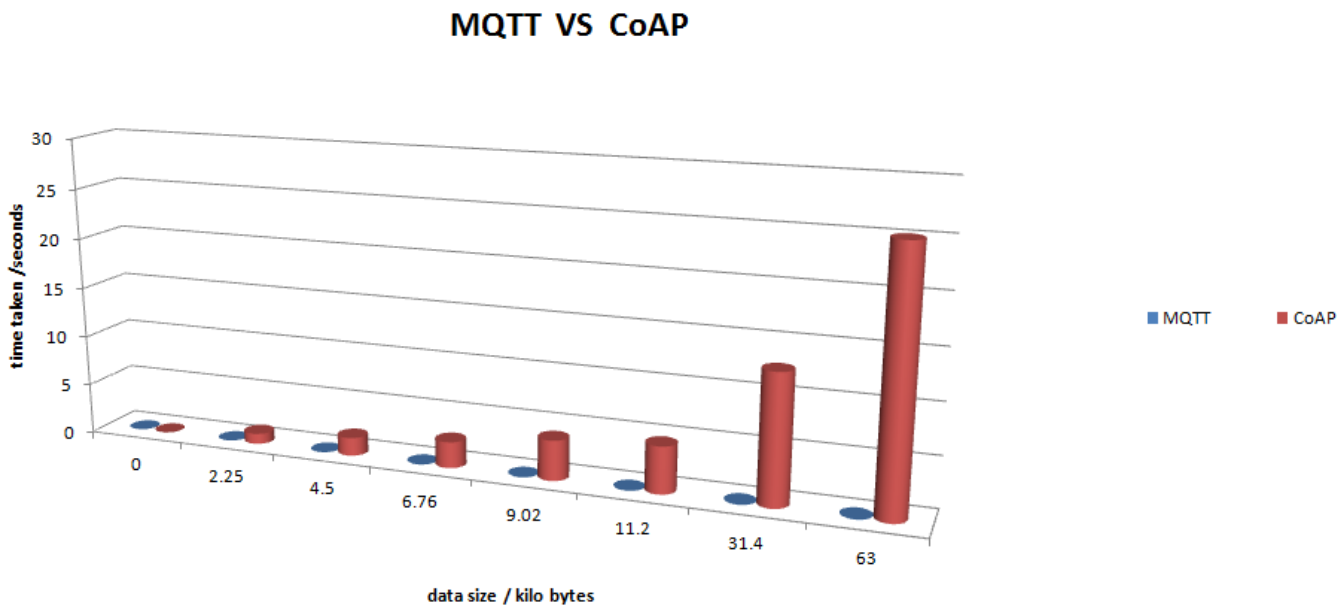

Figure 9: Time consuming in MQTT-SN and CoAP protocols with different data file transmitted.

4.2 Power Consumption Experiment

Similarly, 7 data file were transmitted and tested to get the power consumption with applying both MQTT-SN and CoAP protocols on Rasberry Pi. In the following table (2), Oscilloscope Reading Mean Voltage(mV) is the voltage across the 1.5 Ohm resistor measured by the Oscilloscope. The Voltage of Raspberry Pi (V) is measured by a digital multimeter. Power Consumption of Raspberry Pi is calculated by applying formula (1) and formula (2).

	Data Size/Kbit	Entries	Oscilloscope Reading Mean Voltage(mV)	Voltage of Raspberry Pi(V)	Power Consumption of Respberry Pi(W)
MQTT-SN	0.00	0.00	85.35	5.01	0.28507
	2.25	35.00	85.21	5.00	0.28403
	4.50	70.00	85.64	4.98	0.28432
	6.76	115.00	84.25	5.02	0.28196
	9.02	140.00	85.11	5.01	0.28427
	11.20	175.00	85.92	5.02	0.28755
	31.40	500.00	82.90	4.99	0.27578
	63.00	1000.00	87.17	5.00	0.29057
CoAP	0.00	0.00	86.15	4.98	0.28602
	2.25	35.00	84.32	5.03	0.28275
	4.50	70.00	85.31	5.01	0.28494
	6.76	115.00	85.11	5.01	0.28427
	9.02	140.00	84.23	5.00	0.28077
	11.20	175.00	85.02	5.02	0.28453
	31.40	500.00	85.13	4.99	0.28320
	63.00	1000.00	84.17	5.02	0.28169

Table 2: Power Consumption of MQTT-SN and CoAP protocols with different data file transmitted.

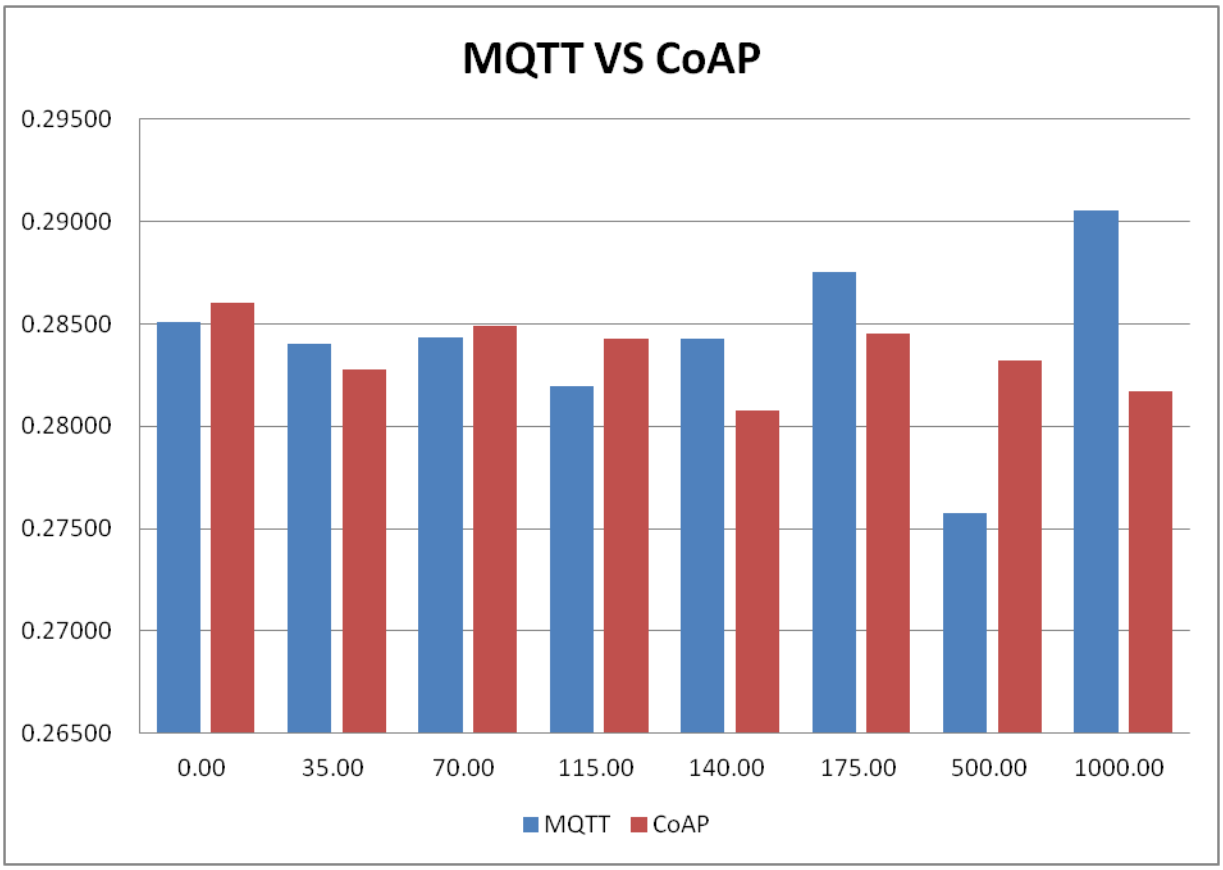

Figure 10: Power Consumption of MQTT-SN and CoAP protocols with different data file transmitted.

CHAPTER 5
DISCUSSION OF RESULTS

The results generated from the Time efficiency experiment indicate that MQTT-SN protocol is more efficient at transmitting data. For each size of data transmitted in both protocols, the server of MQTT-SN spent less time than the server of CoAP to complete the transmission. As we can see from the table (1) and figure (1), the size of data started at 2.25 Kbit where it toke 0.00068 seconds to transmit by using MQTT-SN protocol and took 1.032113 seconds by CoAP protocol. The time consumed by MQTT-SN protocol is around 1,545 times less than the CoAP protocol. As the data size increases, MQTT-SN protocol is providing a more efficient data transmission at around 1,000 times faster than CoAP protocol. .

In the Power Consumption experiment, results are more complicated. The MQTT-SN client (Raspberry Pi) has a power consumption of 0.28507 W which is close to the power consumption of CoAP client (Raspberry Pi) 0.28602 W before transmitting data. As both protocols' client start transmitting data, the power consumption of these clients are changing randomly. Based on these results, it is hard to tell the relationship between the size of data file and the power consumption in one protocol, as well as the difference between the two protocols. This result does not satisfy the assumption of this experiment where MQTT-SN is more power efficient than CoAP [10]. For this problem, one of the concern is that transmitting data in a protocol has a much less effect on the power consumption of Raspberry Pi than other undergoing processes which is difficult to be monitored and controlled. Another cause of this failure could be that the power consumption of both protocols are too low to be measured and displayed by the Oscilloscope.

CHAPTER 6

CONCLUSION

In a summary, MQTT-SN protocol shows a better performance in time efficiency of transmitting data comparing to CoAP protocol. This advantage of MQTT-SN protocol is beneficial for WSN where data are frequently transmitted. To better satisfied WSN, further experiments on MQTT-SN will be needed, such as the security and loss rate of the protocol.

From the results of Power Consumption experiment, it is difficult to clarify which one of the two protocols has a better performance. After tests of transmitting different data file in both MQTT-SN and CoAP protocols, the two protocols' specific influence on power consumption of the Raspberry is still unclear. One of the concern for this failure is that transmitting data in a protocol has a much less effect on the power consumption of Raspberry Pi than other undergoing processes which is difficult to be monitored and controlled. Also, the power consumption of both protocols are too low to be measured and displayed by the Oscilloscope could be another cause of the failure. Therefore, the experiment on Power Consumption of MQTT-SN protocol and CoAP need to be improved and re-preformed before the more power-efficient protocol can be decided.

BIBLIOGRAPHY

[1] F. L. Lewis, Wireless Sensor Networks,

https://pdfs.semanticscholar.org/738d/810dbcab94fe2911dead4666260653f84dc6.pdf

[2] Urs Hunkeler, Hong Linh Truong, Andy Stanford-Clark, MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks, 2008

http://ieeexplore.ieee.org.umiss.idm.oclc.org/document/4554519/

[3] Performance evaluation of MQTT and CoAP via a common middleware, 2014

http://ieeexplore.ieee.org.umiss.idm.oclc.org/document/6827678/

[4] The X-Patch: The world's most widely deployed wearable head Impact Monitor. http://x2biosystems.com/x-patch-pro-wearable-sensor/

[5] MQTT For Sensor Networks (MQTT-SN) Protocol Specification, 2013

http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf

[6] End-to-end service assurance in IoT MQTT-SN, 2015

http://ieeexplore.ieee.org.umiss.idm.oclc.org/document/7157991/

[7]Z. Shelby, Sensinode, K. Hartke, C. Bormann, and U. B. TZI, “Constrained application protocol (coap) draft-ietf-core-coap-17,”

https://tools.ietf.org/html/rfc7252

[8] Information Science Institute, University of Southern California, "Transmission Control Protocol," 1981

https://www.ietf.org/rfc/rfc793.txt

[9] J. Postel, "User Datagram Protocol," 1980

https://tools.ietf.org/html/rfc768

[10] M. H. Amaran, N. A. M. Noh, M. S. Rohmad, H. Hashim, "A Comparison of Lightweight Communication Protocols in Robotic Applications," Procedia Computer Science, 2015

http://www.sciencedirect.com/science/article/pii/S1877050915038193

[11] Z. Shelby, "Constrained Application Protocol (CoAP) Tutorial," April, 2014

https://www.youtube.com/watch?v=4bSr5x5gKvA

[12] American Association of Neurological Surgeons, "Sports-related Head Injury," August 2014

http://www.aans.org/patient%20information/conditions%20and%20treatments/sports-related%20head%20injury.aspx

[13] The Eclipse Foundation, "Mosquitto," 2017
http://www.eclipse.org/proposals/technology.mosquitto/

[14] SwitchDoc Labs, " Tutorial: IOT / Installing and Testing Mosquitto MQTT on the Raspberry Pi,"

http://www.switchdoc.com/2016/02/tutorial-installing-and-testing-mosquitto-mqtt-on-raspberry-pi/

[15] L. F. Rahman, "Tutorial on txThings (CoAP Libraries)," Eindhoven University of Technology, 2016-2017

http://www.win.tue.nl/~lrahman/iot_2016/tutorial/txThings_2016.pdf

[16] "Wireshark on Raspberry Pi Tutorial,"
http://donsthinktank.blogspot.com/2015/07/wireshark-raspberry-pi.html